LPORAN JOBSHEET 14

TREE

Mata Kuliah : Algoritma dan Struktur Data

Dosen : **Mungki Astiningrum, S.T., M.Kom.**

Ilham Dharma Atmaja
244107020220
Kelas : 1A
Absen : 14

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG TAHUN 2025

1.1 Tujuan Praktikum

Setelah melakukan praktikum ini, mahasiswa mampu:

1. memahami model Tree khususnya Binary Tree
2. membuat dan mendeklarasikan struktur algoritma Binary Tree.
3. menerapkan dan mengimplementasikan algoritma Binary Tree dalam kasus Binary Search Tree

1.2 Praktikum 1

1) Class Mahasiswa

```
public class Mahasiswa14 {

    String nim;

    String nama;

    String kelas;

    double ipk;


    public Mahasiswa14(){

    }


    public Mahasiswa14(String nim, String nama, String kelas, double ipk) {

        this.nim = nim;

        this.nama = nama;

        this.kelas = kelas;

        this.ipk = ipk;

    }

    public void tampil() {

        System.out.println("NIM    : " + this.nim + " " +

        "Nama   : " + this.nama + " " +

        "Kelas  : " + this.kelas + " " +

        "IPK    : " + this.ipk);

    }

}
```

2) Class Node

```
public class Node14 {
    Mahasiswa14 mahasiswa;
    Node14 left, rigth;

    public Node14() {
    }

    public Node14(Mahasiswa14 mahasiswa) {
        this.mahasiswa = mahasiswa;
        this.left = null;
        this.right = null;
    }
}
```

3) Class binary tree

```
public class BinaryTree14 {
   Node14 root;

   public BinaryTree14() {
      root = null;
   }

   public boolean isEmpty() {
      return root == null;
   }
   public void add(Mahasiswa14 mahasiswa) {
      Node14 newNode = new Node14(mahasiswa);
      if (isEmpty()) {
         root = newNode;
      } else {
         Node14 current = root;
         Node14 parent;
```

```
        while (true) {
            parent = current;
            if (mahasiswa.ipk < current.mahasiswa.ipk) {
                current = current.left;
                if (current == null) {
                    parent.left = newNode;
                    return;
                }
            } else {
                current = current.right;
                if (current == null) {
                    parent.right = newNode;
                    return;
                }
            }
        }
    }
}
boolean find(double ipk) {
    boolean result = false;
    Node14 current = root;
    while (current != null) {
        if (current.mahasiswa.ipk == ipk) {
            result = true;
            break;
        } else if (ipk > current.mahasiswa.ipk) {
            current = current.left;
        } else {
            current = current.right;
        }
    }
```

```
    return result;
}
void traversePreorder (Node14 node) {
    if (node != null) {
        traversePreorder(node.left);
        node.mahasiswa.tampil();
        traversePreorder(node.right);
    }
}

void traverseInOrder (Node14 node) {
    if (node != null) {
        traverseInOrder(node.left);
        node.mahasiswa.tampil();
        traverseInOrder(node.right);
    }
}

void traversePostOrder (Node14 node) {
    if (node != null) {
        traversePostOrder(node.left);
        traversePostOrder(node.right);
        node.mahasiswa.tampil();
    }
}

Node14 getSuccessor(Node14 del ) {
    Node14 successsor = del.right;
    Node14 successsorParent = del;

    while (successsor.left != null) {
```

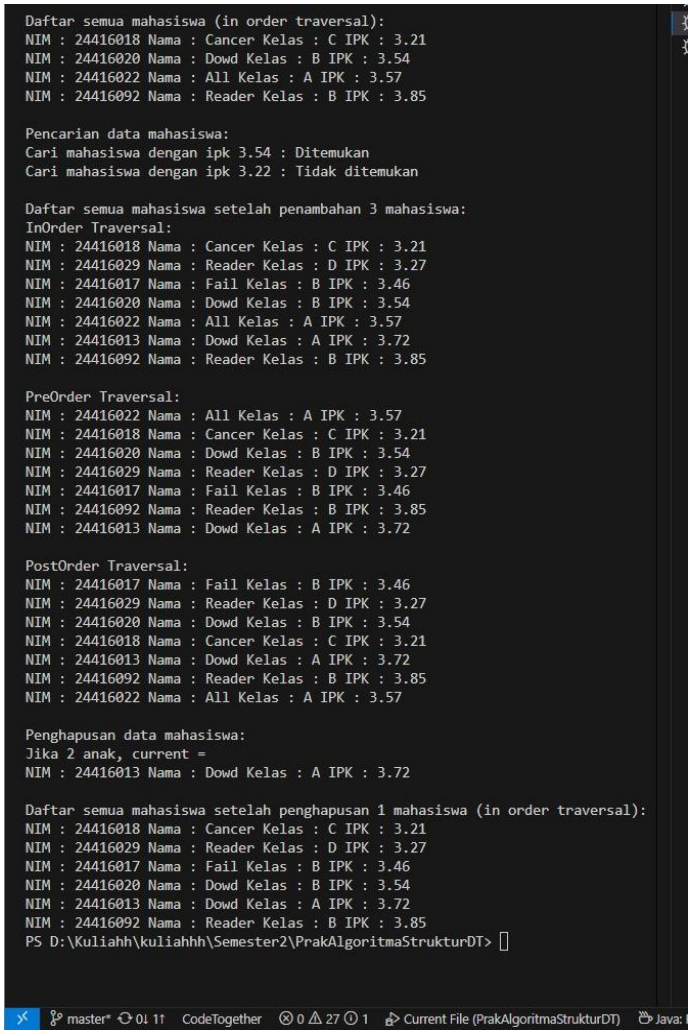
```
            successsorParent = successsor;
            successsor = successsor.left;
        }

        if (successsor != del.rigth) {
            successsorParent.left = successsor.rigth;
            successsor.rigth = del.right;
        }
        return successsor;
    }
    void delete(double ipk) {
        if (isEmpty()) {
            System.out.println("Binary tree kosong");
            return;
        }
        // mencari node yang akan dihapus
        Node14 parent = root;
        Node14 current = root;
        boolean isLeftChild = false;
        while (current  != null) {
            if(current.mahasiswa.ipk == ipk) {
                break;
            } else if (ipk < current.mahasiswa.ipk) {
                parent = current;
                current = current.left;
                isLeftChild = true;
            } else if (ipk > current.mahasiswa.ipk) {
                parent = current;
                current = current.right;
                isLeftChild = false;
            }
```

```
    }
    //penghapusan node
    if (current == null) {
        System.out.println("Data tudak ditemukan");
        return;
    }else{
        //jikatidak ada anak (leaf), maka node dihapus
        if (current.left == null && current.right == null) {
            if (current == root) {
                root = null;
            }else{
                if (isLeftChild) {
                    parent.left = null;
                } else {
                    parent.right = null;
                }
            }
        }else if(current.left == null){
            //jika hanya ada anak kanan
            if (current == root ) {
                root = current.right;
            } else {
                if (isLeftChild) {
                    parent.left = current.right;
                } else {
                    parent.right = current.right;
                }
            }
        }else if(current .rigth == null){
            //jika hanya ada anak kiri
            if (current == root) {
```

```
                root = current.left;
            } else {
                if (isLeftChild) {
                    parent.left = current.left;
                } else {
                    parent.rigth = current.left;
                }
            }
        }else{
            //jika ada dua anak
            Node14 successor = getSuccessor(current);
            System.out.println("Jika ada 2 anak, current = ");
            successor.mahasiswa.tampil();
            if (current == root) {
                root = successor;
            } else {
                if (isLeftChild) {
                    parent.left = successor;
                } else {
                    parent.rigth = successor;
                }
            }
            successor.left = current.left;
        }
    }
  }
}
```

1.3. Verifikasi



1.4. Tugas

1. data digunakan untuk menyimpan elemen-elemen (node) dari pohon biner dalam bentuk array.idxLast menyimpan indeks terakhir yang berisi data pada array, sehingga memudahkan dalam menambah atau mengetahui jumlah elemen pada pohon.
2. Method ini digunakan untuk mengisi atau menginisialisasi array data dengan elemen-elemen pohon biner, biasanya dari input atau data yang sudah ada.
3. Method ini digunakan untuk melakukan penelusuran (traversal) pohon biner secara in-order, yaitu mengunjungi node kiri, node saat ini, lalu node kanan secara berurutan.
4. Left child: indeks 2*2+1 = 5
   Right child: indeks 2*2+2 = 6
5. Statement ini digunakan untuk menandai bahwa elemen terakhir yang berisi data pada array berada di indeks ke-6, sehingga traversal atau operasi lain hanya dilakukan sampai indeks tersebut.
6. Indeks tersebut digunakan untuk menentukan posisi left child (2*idxStart+1) dan right child (2*idxStart+2) dari suatu node pada array.

2.1. Praktikum 2

1) Class BinaryTreeArray

```
public class BinaryTreeArray14 {
    Mahasiswa14[] dataMahasiswa;
    int idxLast;

    public BinaryTreeArray14() {
        this.dataMahasiswa = new Mahasiswa14[10];

        this.idxLast = -1;
    }

    void populateData(Mahasiswa14 dataMhs[], int idxLast) {
        this.dataMahasiswa = dataMhs;
        this.idxLast = idxLast;
    }

    void traverseInOrder(int idxStart) {
        if (idxStart <= idxLast) {
            if (dataMahasiswa[idxStart] != null) {
                traverseInOrder(2 * idxStart + 1);
                dataMahasiswa[idxStart].tampil();
                traverseInOrder(2 * idxStart + 2);
            }
        }
    }
}
```

2) Class Main

```
public class BinaryTreeArrayMain14 {
    public static void main(String[] args) {
        BinaryTreeArray14 bta = new BinaryTreeArray14();

        Mahasiswa14 mhs1 = new Mahasiswa14("244160121", "Ali", "A", 3.57);
        Mahasiswa14 mhs2 = new Mahasiswa14("244160185", "Candra", "C", 3.41);
        Mahasiswa14 mhs3 = new Mahasiswa14("244160221", "Badar", "B", 3.75);
```

```
        Mahasiswa14 mhs3 = new Mahasiswa14("244160221", "Badar", "B", 3.75);
        Mahasiswa14 mhs4 = new Mahasiswa14("244160220", "Dewi", "B", 3.35);

        Mahasiswa14 mhs5 = new Mahasiswa14("244160131", "Devi", "A", 3.48);
        Mahasiswa14 mhs6 = new Mahasiswa14("244160205", "Ehsan", "D", 3.61);
        Mahasiswa14 mhs7 = new Mahasiswa14("244160170", "Fizi", "B", 3.86);

        Mahasiswa14[] dataMahasiswas = { mhs1, mhs2, mhs3, mhs4, mhs5, mhs6,
mhs7,
        null, null, null };
        int idxLast = 6;
        bta.populateData(dataMahasiswas, idxLast);
        System.out.println("\nInorder Traversal Mahasiswa:");

        bta.traverseInOrder(0);

        bta.add(mhs1);
        bta.add(mhs2);
        bta.add(mhs3);
        bta.add(mhs4);
        bta.add(mhs5);
        bta.add(mhs6);
        bta.add(mhs7);

        // Tampilkan hasil traversal
        System.out.println("\nInorder Traversal Mahasiswa:");
        bta.traverseInOrder(0);

        System.out.println("\nPreorder Traversal Mahasiswa:");
        bta.traversePreOrder(0);

    }
}
```

2.2. Pertanyaaan

1. data adalah array yang digunakan untuk menyimpan elemen-elemen (node) dari pohon biner.
   idxLast adalah indeks terakhir yang berisi data pada array, sehingga dapat mengetahui jumlah elemen yang ada di pohon.
2. Method ini digunakan untuk mengisi atau menginisialisasi array data dengan elemen-elemen pohon biner
3. Method ini digunakan untuk melakukan penelusuran (traversal) pohon biner secara in-order, yaitu mengunjungi node kiri, node saat ini, lalu node kanan secara berurutan.
4. Left child: indeks 2*2+1 = 5
   Right child: indeks 2*2+2 = 6
5. Untuk menandai bahwa elemen terakhir yang berisi data pada array berada di indeks ke-6, sehingga operasi pada pohon hanya dilakukan sampai indeks tersebut.
6. **2*idxStart+2 dalam rekursif:**
   Karena pada struktur pohon biner yang disusun dalam array, anak kiri dari node di indeks i berada di indeks 2i+1 dan anak kanan di indeks 2i+2. Rumus ini digunakan agar traversal atau operasi rekursif dapat berjalan sesuai struktur pohon biner dalam array.

3. TUGAS

```
// TUGAS 1
  public void addRekursif(Mahasiswa14 mahasiswa) {
      root = tambahRekursif(root, mahasiswa);
  }

  private Node14 tambahRekursif(Node14 current, Mahasiswa14 mahasiswa) {
      if (current == null) {
          return new Node14(mahasiswa);
      }
      if (mahasiswa.ipk < current.mahasiswa.ipk) {
          current.left = tambahRekursif(current.left, mahasiswa);
      } else {
          current.right = tambahRekursif(current.right, mahasiswa);
      }
      return current;
  }
```

```
    // TUGAS 2
    public void cariMinIPK() {
        if (root == null) {
            System.out.println("Tree kosong.");
        } else {
            Node14 current = root;
            while (current.left != null) {
                current = current.left;
            }
            System.out.println("Mahasiswa dengan IPK terkecil:");
            current.mahasiswa.tampil();
        }
    }

    public void cariMaxIPK() {
        if (root == null) {
            System.out.println("Tree kosong.");
        } else {
            Node14 current = root;
            while (current.right != null) {
                current = current.right;
            }
            System.out.println("Mahasiswa dengan IPK terbesar:");
            current.mahasiswa.tampil();
        }
    }

    // TUGAS 3
    public void tampilMahasiswaIPKdiAtas(double ipkBatas) {
        tampilMahasiswaIPKdiAtasRekursif(root, ipkBatas);
    }

    private void tampilMahasiswaIPKdiAtasRekursif(Node14 node, double ipkBatas)
{
        if (node != null) {
```

```
            tampilMahasiswaIPKdiAtasRekursif(node.left, ipkBatas);
            if (node.mahasiswa.ipk > ipkBatas) {
                node.mahasiswa.tampil();
            }
            tampilMahasiswaIPKdiAtasRekursif(node.right, ipkBatas);
        }
    }
```

```
    // TUGAS SOAL NO 4
     void add(Mahasiswa14 data) {
         if (idxLast >= dataMahasiswa.length - 1) {
             System.out.println("Array penuh, tidak dapat menambahkan data.");
         } else {
             idxLast++;
             dataMahasiswa[idxLast] = data;
         }
     }

     void traversePreOrder(int idxStart) {
         if (idxStart <= idxLast) {
             if (dataMahasiswa[idxStart] != null) {
                 dataMahasiswa[idxStart].tampil();
                 traversePreOrder(2 * idxStart + 1);
                 traversePreOrder(2 * idxStart + 2);
             }
         }
     }
```

3.2 LINK GIT HUB

https://github.com/ianmen10/SEMESTER-genap2.git